Extending Enterprise and Domain Engineering Architectures to Support the Object Oriented Paradigm

Fred A. Maymir-Ducharme, PhD
Lockheed Martin, Mission Systems
fred.a.maymir-ducharme@lmco.com

1.0 BACKGROUND As the size and complexity of software systems increase and budgets decrease, the U.S. Government has realized the dire need to provide guidance to develop systems more effectively and efficiently. We can no longer afford to "reinvent the wheel" every time a new system is needed. Engineering families of systems, product lines, and exploiting commercial off-the-shelf (COTS) software and Government off-the-shelf (GOTS) software are just a few approaches to achieving better engineered systems. In addition, software intensive systems must be able to work together and exchange information. While interoperability is important for many information systems, it is essential for military systems, which must be capable of supporting lifesaving operations that may require changing a mix of forces, at a moment's notice, just about anywhere in the world.

U.S. Government has developed architectural guidance and policy to achieve the required interoperability, as well as engineering systems faster, better and cheaper. These initiatives and products only address "what" should be done. Program managers and systems engineers tasked to deliver these systems depend on technology addressing the associated "how to's." This paper addresses the technology (concepts, processes, methods and tools) used on multiple programs to effectively and efficiently engineer military systems, using various architecture guidance, policy and products. One of the major themes (i.e., lessons learned) of this paper is that there are many conflicts between the technologies associated with Object Oriented approaches and the more traditional Structured/Functional approaches. If both approaches are used by an organization, these challenges must be identified early and dealt with accordingly.

2.0 DISCIPLINED SOFTWARE ENGINEERING The way we engineer our systems is continuously changing and improving. We can no longer treat each new project as a single, new and independent development effort and not build on previous engineering efforts and experience. Instead we need to view these systems within the context of similar systems built in the past, exploiting the commonalities and engineering the appropriate variances. Additionally, we must leverage off existing reusable assets and develop new ones with reuse in mind. Reuse is an integral part of a disciplined software engineering practice, which is continuously improving its technology/asset base and processes. In order to meet today's software challenges [6] of increasing demand, complexity and size, we need to establish new ways of fusing together information about what assets exist and need to be woven into the processes used to guide our engineering activities. Various software engineering methods, processes and tools exist to help take advantage of available information about data, process, and software assets needed to make the engineering decisions governing the quality of the products that evolve as a consequence of their mechanization.

Disjoint engineering efforts (i.e., Information Engineering, Domain Engineering and Application Engineering) result in engineering process stovepipes. Each engineering level develops models representing the associated requirements. Each engineering practice designs a solution (sometimes captured by an architecture or design). And each engineering practice then implements/develops their products. The challenge is to fuse these engineering methods (and thereby their work products) to eliminate redundancies, inconsistencies and other anomalies. The goal is to define and implement a disciplined software engineering practice that assures that the work products and standards produced any phase of the lifecycle are consistent and coordinated with the work products and standards of all associated lifecycle phases. For example, data models developed during the enterprise modeling phases must feed into the appropriate domain engineering and application engineering phases; and reciprocally, provide feedback to the enterprise efforts when the data models need to be modified or extended. Applications developed individually without considering common and/or related systems in the domain result in stovepipe systems /

applications. Likewise, domains engineered without considering the broader enterprise (e.g., common data elements, business functions, the need to interoperate, etc.), can result in stovepipe domains.

Mature engineering disciplines support clear separation of routine problem solving from R&D. These disciplines have publicly-held, experience based, and formally transmitted technology bases that include product models (e.g., designs, specifications, performance ranges) and practice models (tools and techniques to apply to the product models) (See Figure 1 below). Furthermore, the qualities of products built from these models are well-understood and predictable before the products are produced.

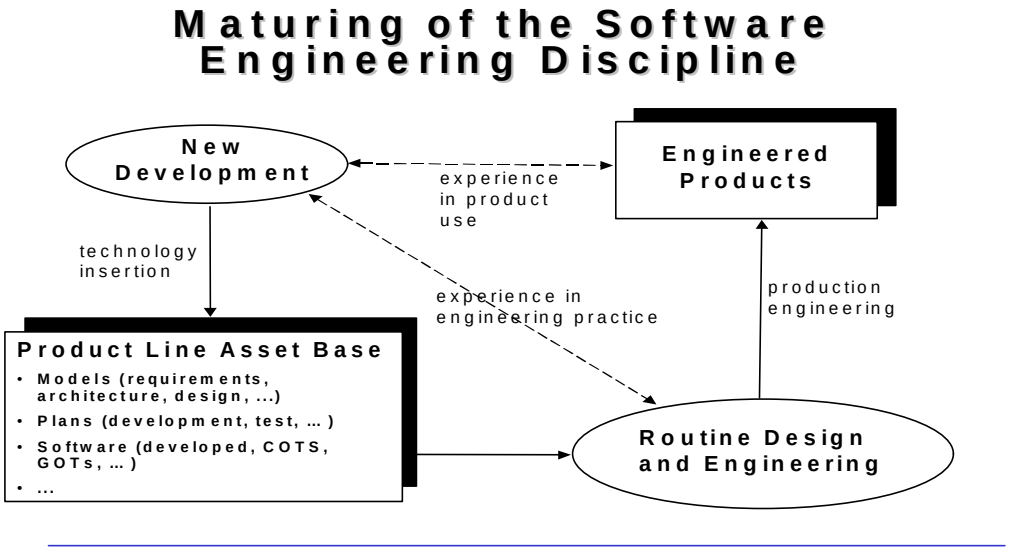

Figure 1 The Maturing of the Software Engineering Discipline

The state-of-the-practice of software engineering is not yet at this level of maturity. Instead of basing new development on a technology base of well-understood models, current software engineering practice tends to start each new application development from scratch with the specification of requirements, and moves directly into the design and implementation. By contrast, this effort's vision of a mature software engineering discipline, as illustrated in the figure above, relies on a technology base of reusable assets and clearly separates routine systems development (i.e., application engineering) from development of the domain-specific technology base (i.e., domain engineering). This separation highlights the need and significance of developing reusable corporate assets including requirements, models, architectures, processes, and components. The application engineering function can then focus on validating and using this technology base, instead of beginning with a blank sheet. In addition to creating the initial set of domain assets, domain engineering processes will continue to add and enhance the technology base according to the requirements associated with application engineering.

Under the USAF Comprehensive Approach to Reusable Defense Software (CARDS) Partnerships Program [20], LM developed and applied, the AF/CARDS Engineered Software (ACES) methodology [21,22,23] (illustrated below), an approach that combines Information Engineering with Domain Engineering and the Object Modeling Technique (OMT). The CARDS Tri-Lifecycle Software Engineering model [1,2,27] (Figure 2 below), reflects three types of engineering activities during the acquisition and life cycle development and maintenance of software intensive systems: Enterprise Engineering [2,3,26], Domain Engineering [23,24.25], and Application Engineering [1,5,7,8,17]. Due to the complexity of engineering all of the systems within the enterprise, as well as the numerous methodologies available for each engineering area, it is likely that information will be lost, regenerated, or not seen as relevant to previous or succeeding activities -- thereby causing redundant work efforts, data and function anomalies, and higher development and maintenance costs. This lack of coordination and communication across processes has been coined

"stovepipe processes" and is analogous to the systems stovepipes dilemma, where systems fail to leverage common data and the necessary interoperability. Approaching the problem with planning oversight of all three activities ensures that information flows from one activity to the next.

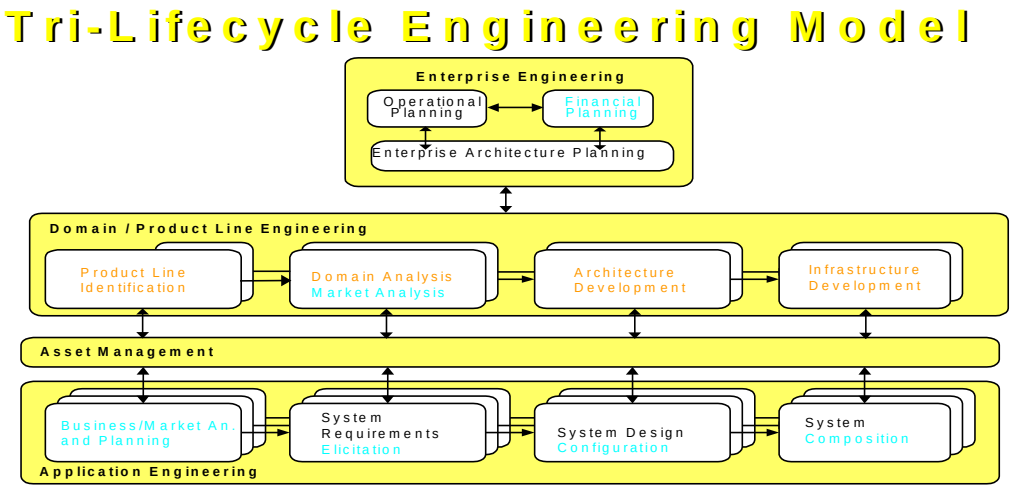

Figure 2 The CARDS Tri-Lifecycle Engineering Model

There are numerous Domain Engineering methods and processes. The primary domain analysis methods (primary because of their validation/applications on various efforts and associated publications) include: Organization Domain Modeling (ODM) [18], a well defined and comprehensive method; Domain Engineering Process (DEP) [27], an extension of object-oriented methods; the SEI Feature Oriented Domain Analysis (FODA) [2] method, considered to be the most mature DE methodology; and SPC's Synthesis [19].

3.0 ARCHITECTURE GUIDANCE U.S. Government guidance and policy such as the Command Control Communications Computers Intelligence Surveillance and Reconnaissance (C4ISR) Architecture Framework, Joint Technical Architecture (JTA), Defense Information Infrastructure (DII) Common Operating Environment (COE) and other US DoD architectural guidance are crucial to achieving interoperability, while building systems faster, better and cheaper.

The JTA [30] is the DoD's specification for interoperability between all DoD systems. Figure 3 below illustrates the relationship of the JTA to other DoD architecture guidance and initiatives. The JTA is based on the Technical Architecture Framework for Information Management (TAFIM), Adopted Information Technology Standards (AITS) – Volume 7 of the TAFIM [12]; and uses the DoD Technical Reference Model (TRM, TAFIM Vol 2) as it's structure for specifying interoperability for each major service area. The JTA defines the service areas, interfaces, and standards (JTA elements) applicable to all DoD systems, and its adoption is mandated for the management, development, and acquisition of new or improved systems throughout DoD. The JTA is complementary to and consistent with other DoD programs and initiatives aimed at the development and acquisition of effective, interoperable information systems -- including the DoD's Specification and Standards Reform, the Information Technology Management Reform Act (ITMRA); DoD C4ISR Architecture Framework, the DoD TRM; the Defense Information Infrastructure Common Operating Environment (DII COE); and Open Systems Initiative.

DOD Architecture Efforts

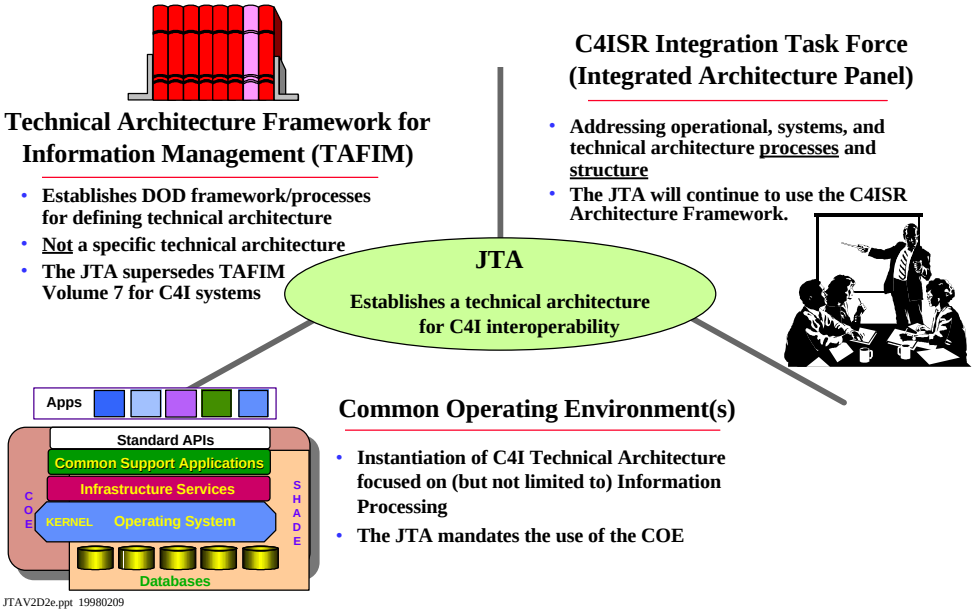

Figure 3 DoD Architecture Guidance

The DoD TRM originated from the TAFIM and was developed to show which interfaces and content needed to be identified. The TRM Working Group (TRMWG) has extended the scope of the TRM to include real-time systems (e.g., weapon systems) and is coordinated with the JTA. As figure 3 indicates, the JTA is also very closely coupled with the DII COE [32] and the C4ISR Architecture Framework [31]. The DII COE is the DoD's implementation of a technical architecture supporting interoperability, supplemented by various common services / utilities to maximize reuse across multiple systems. And as the figure below indicates, the JTA is one of the three architectures defined by the C4ISR Architecture Framework.

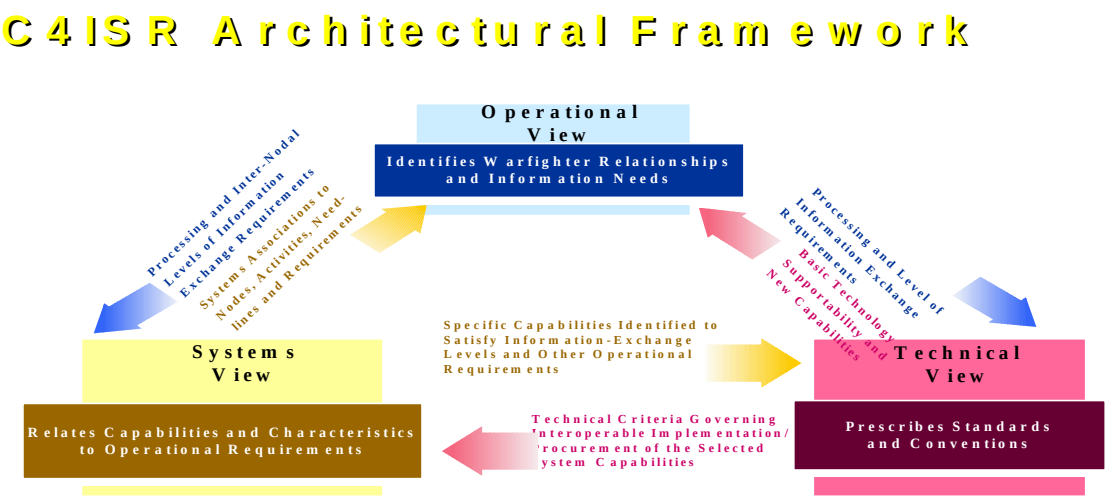

Figure 4 DoD C4ISR Architecture Framework

4.0 LESSONS LEARNED Lockheed Martin (LM) worked with the USAF to replace existing transportation information systems. These systems were designed as stand-alone applications serving individual offices or functions. The resulting system gaps and overlaps, and the concomitant data and process redundancy and inconsistency, have caused problems for both information users and systems maintainers. USAF's goal is to reduce development and maintenance costs while enhancing support to the warfighter. Its objectives are to develop a unified transportation system and environment -- consisting of a corporate database, corporate applications, common functionality, and a corporate network. The strategy for reaching these objectives is to introduce a reuse-based approach to application systems development. The approach is to replace stovepipe information systems with a set of integrated applications that cut across organizational and functional lines and to implement a virtual corporate database. The corporate database will appear to the user to be integrated and monolithic but will actually be composed of physically distributed, heterogeneous databases and - for the foreseeable future - legacy USAF and DoD systems.

The USAF employed the Zachman Framework to guide its Information Systems Architecture development. Within this framework, USAF addressed its enterprise-wide data integration objectives by applying Steven Spewak's Enterprise Architecture Planning (EAP) process (an Information Engineering (IE) technique). The product, a high-level Transportation System Master Plan, includes a Mission Analysis, Information Architecture, Application Architecture, and Implementation Plan.

ACES was based on the CARDS Tri-Lifecycle Engineering Model, which extended the DARPA Software Technology for Adaptable, Reliable Systems (STARS) Dual Lifecycle Model (i.e., Domain Engineering) to include Information/Enterprise Engineering. The complete ACES methodology addresses Enterprise Engineering (e.g., Spewak's EAP) [3], Object-Oriented (OO) Domain Engineering, and OO Applications Engineering (using Rumbaugh's OMT) [1]. The transition from Enterprise Engineering to Domain Engineering uses IE-based affinity analysis between data entities and business processes to identify and scope candidate domains. It then uses an OO approach to analyze inter-domain relationships in terms of service requests. Within each domain of focus, ACES uses FODA to identify and categorize reuse opportunities, and OMT to develop reusable business objects that satisfy semantic information integration and synthesis requirements. Application Engineering consists of matching specific user requirements to business objects and developing the necessary application-specific objects.

There were many lessons learned throughout this effort. Transitioning from the very functional (sometimes referred to as "structured") information/enterprise engineering methods to an OO solution incurred several challenges. Applying affinity analyses and multi-domain modeling techniques over the enterprise information element lifecycles to scope the domains and hence group the service objects proved to be key in this transition. The fundamental differences between structured and OO approaches must be considered in the many translations and transitions across the various methods and workproducts within the Tri-Lifecycle. The Data Access Layer within the framework in Figure 4 below was necessary to deconflict data access between the structured legacy code and the new OO code. The figure below summarizes the integration and application of DoD architectural guidance / products with the associated architecture technology. Lessons learned will be discussed during the panel session. Additional lessons learned in applying the ACES methodology, based on the CARDS Tri-Lifecycle Engineering Model above are discussed in the references listed below. The figure below illustrates the integration of both, the technology (e.g., EAP, ACES, OMT) and the DoD guidance/products (e.g., C4ISR Architecture Framework, JTA and COE) used to reengineer the USAF's Defense Transportation System.

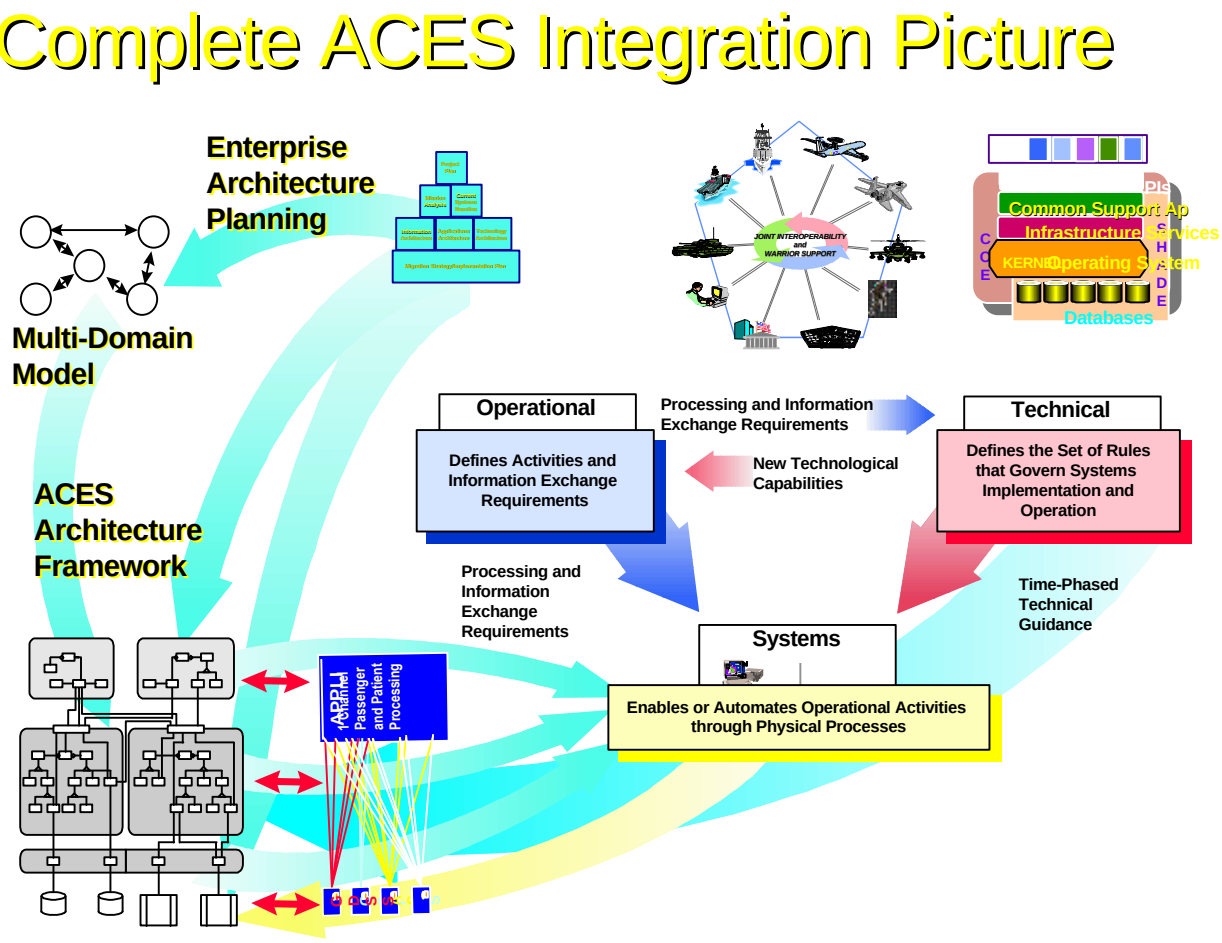

Figure 5 ACES'Integration of Architecture Guidance, Policy & Technology

5.0 ACKNOWLEDGMENTS Special thanks and credit are due to the teams that worked with me on the development and application of these technologies, as well as the team supporting the development of the DoD JTA (2.0). These include Jim Fulton, Mike Webb, Robin Burdick, Frank Svoboda, Roger Whitehead, David Weisman, Lucy Haddad, Nancy Solderitsch, Paul Kogut, Wil Berrios, Russ Richards, Olimpia Velez, Jim DeGoey, Mark Dowson and many others.

6.0 REFERENCES

1. J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. Object-Oriented Modeling and Design, Prentice-Hall, 1991.
2. K. Kang, S. Cohen, J. Hess, W. Novak, and S. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study, CMU/SEI-90-TR-21, Carnegie-Mellon University, Software Engineering Institute, November 1990.
3. S. Spewak. Enterprise Architecture Planning, John Wiley & Sons, 1992
4. J. Zachman. "A Framework for Information Systems Architecture," IBM Systems Journal, Vol. 26, No. 3, 1987.
5. W. Royce, "Managing the Development of Large Software Systems: Concepts & Techniques" 1970
6. OUSD/AT "Defense Report of the Defense Science Board Task Force on Military Software," 1987
7. B. Boehm, "A Spiral Model of Software Development an Enhancement," 1988.
8. W. Royce, "TRW's Ada Process Model for Incremental Development of Large Software Systems," 1990.
9. MIL-STD-498 "Software Development and Documentation" 1994
10. EIA/IEEE J-STD-016 "Software Life-Cycle Processes" 1995
11. ISO/IEC STD 12207 "IT – Software Life-Cycle Processes" 1995
12. DoD, "Technical Architecture Framework for Information Management (TAFIM)" Version 2.0, Defense Information Systems Agency, Center for Architecture, June 1994
13. The DoD Enterprise Model, Volume I: Strategic Activity and Data Models, Office of the Secretary of Defense, ASD (C3I), January 1994.

14. The DoD Enterprise Model, Volume II: Using the DoD Enterprise Model, A Strategic View of Change in DoD, A White Paper, Office of the Secretary of Defense, ASD (C3I), January 1994.
15. Information Management Program, DoD Directive 8000.1, October 1992.
16. DoD Data Administration, DoD Directive 8320.1, September 1991.
17. IEEE Standard for Developing Software Life Cycle Processes, IEEE Computer Society, IEEE STD 1074-1991, January 1992.
18. Simos, M., "ARPA STARS Organization Domain Modeling (ODM) Guidebook Version 1.0" March 1995
19. "Synthesis, A Reuse-Based Software Development Methodology, Process Guide, Version 1.0," Software Productivity Consortium, October 1992.
20. Maymir-Ducharme, F.A., Weisman, D. "A.F./CARDS Technology Transition Program: Reuse Partnerships," proceedings of the Reuse '95 Workshop, August 1995.
21. Maymir-Ducharme, F.A., "Variant Domain Engineering Approaches," proceedings of the Workshop on Institutionalizing Software Reuse WISR'95, July 1995.
22. Maymir-Ducharme, F.A., Svoboda, F. "Translating Enterprise Models into Domain Engineering Workproducts," Proceedings of the Reuse '96 Workshop, August 1996.
23. Maymir-Ducharme, F.A., (WG Chair). "Opportunistic, Systematic and Optimized Domain Engineering Approaches" Proceedings of the Reuse '96 Workshop, August 1996.
24. Maymir-Ducharme, F.A., "Product Lines, Just One of Many Domain Engineering Approaches," Proceedings of the NASA Software Reuse Workshop, sponsored by GMU and NASA SORT Program, October 1997,
25. Maymir-Ducharme, F.A., "A Product Line Business Model," Proceedings of ARES'96 (Architectural Reasoning for Embedded Software), sponsored by ESPRIT IV project no. 20.477, Las Navas, Spain, 18-20 Nov. 1996.
26. Martin, James, "Information Engineering : A Trilogy," Prentice Hall, Inc., Englewood Cliffs, NJ 1989.
27. Defense Information Systems Agency (DISA), "Domain Engineering Process (Version 2)" 28 April 1995.
28. Combined Communications Electronics Board (CCEB), "Combined Interoperability Technical Architecture (CITA) Rationale and Development Framework (Ver. 0.2) March, 1998.
29. CCEB, "Combined Interoperability Technical Architecture (CITA), Ver. 0.1," March 1998.
30. DISA, "DOD Joint Technical Architecture (JTA)," http://www-jta.itsi.disa.mil/
31. OSD/C3I "C4ISR Architecture Framework," http://www.cisa.osd.mil/organization/architectures/
32. DISA, "Defense Information Infrastructure (DII) Common Operating Environment (COE)," http://spider.osfl.disa.mil/dii/